TUNA and HEAPO: Suite for Developing Software for NVM

Youjip Won
Hanyang University, Seoul, Korea
Outlines

- TUNA: NVRAM Emulation Platform
- HEAPO: Persistent Heap Layer for Non-Volatile Memory
TUNA: The Platform for Emulating NVM
Non-volatile Memory

- DRAM: Volatile
- NVRAM: Non-volatile, High Capacity, Fast, Byte-addressable
- Flash/HDD: Block addressing, Slow

Flash Memory Summit 2015
Santa Clara, CA
Characteristics of Memory Devices

**DRAM**
- Read: 10-50ns
- Write: 10-50ns
- Energy/bit: 0.005pJ
- Endurance: $10^{16}$
- Density: 6~8F²

**FRAM**
- Read: 110ns
- Write: 150ns
- Energy/bit: 1pJ
- Endurance: $10^{10}$-$10^{13}$
- Density: 6~10F²

**STT-MRAM**
- Read: 10-30ns
- Write: 13-95ns
- Energy/bit: 0.1~2.5pJ
- Endurance: $10^{15}$
- Density: 14-64F²

**PCM**
- Read: 20-70ns
- Write: 50-500ns
- Energy/bit: 2-25pJ
- Endurance: $10^7$-$10^8$
- Density: 8~16F²

**NAND-Flash**
- Read: 25us
- Write: 250us
- Energy/bit: 0.000002pJ
- Endurance: $10^3$-$10^3$
- Density: 4~8F²
Software Layers for NVM

Persistent Heap
- NV-heap (Coburn, UCSD, ASPLOS’11)
- Mnemosyne (Volos, UW, ASPLOS’11)
- SoftPM (Guerra, FIU, Usenix ATC’12)
- WSP (Narayanan, MS, ASPLOS’12)
- HEAPO (Hwang, HYU, ACM TOS’14)
- NVM Duet (Liu, NTU, ASPLOS’14)

Byte-addressable Filesystem
- BPFS (Condit, MS, SOSP’09)
- FRASH (Jung, HYU, ACM TOS’12)
- SCMFS (Wu, Texas A&M, SC’11)
- PMFS (Dulloor, Intel, Eurosys’14)
- PMBD (Chen, LSU, MSST’14)

Fast Block Device
- Moneta (Caulfield, UCSD, MICRO’10)
- Onyx (Akel, UCSD, HotStorage’11)
- PCMSSD (Kim, IBM, FAST’14)
- DC Express (Vucinic, UCSD, FAST’14)
- nvramdisk (Jung, HYU)
Are the performance results credible?

Sources for inaccuracy

- **Real NVM device**: small scale and for dedicated purpose
- **Emulating NVM with DRAM**: DRAM is much faster.
- **Introducing software delay**: code overhead and subsequent cache pollution
- **Cycle-accurate simulation**: Lengthy simulation and cannot simulate realistic scenario
- **NVRAM emulation platform (Intel PMEP)**: Only available in x86, cannot simulate the non-volatility.
Hardware platform for NVM emulation for mobile device

- Variable Latency
  - PCM, STT-MRAM, and etc.

- Non-volatility emulation

- Mobile platform
TUNA: Hardware platform for NVM emulation

- Use DRAM to emulate NVRAM
- For Non-volatility
  - Separate power source for emulated NVRAM
- For Variable latency
  - API for adjusting the NVM latency
  - Separate latencies for load and store
- For Mobile platform
  - ARM
  - Android ported
TUNA Organization

PS-DRAM Component

Xilinx Zynq (XC7Z045)

Processing System (PS, ARM Co-A9)
- Core #1
- Memory interface
- AXI master

Programmable Logic (PL, Xilinx Kintex-7)

- Core #2
- Xilinx DRAM controller
- Delay

*PS is hardwired

Separate power source

PL-DRAM So-DIMM

Flash Memory Summit 2015
Santa Clara, CA
Emulating non-volatility with DRAM

- Separate power control
  - Self-refresh management circuit
  - PS-only reboot (NVM-DRAM under self refresh or auto-refresh)

Based on real measurement on steady state logic value!
Support Variable Latency

- Independent RD/WR latency tuning
- Delay AXI packets from PS
  - Postpone handshake signals on AW/AR channel

\[
\text{Delay( \{ARVALID, ARREADY\}, read\_lat);}
\]
\[
\text{Delay( \{AWVALID, AWREADY\}, write\_lat);}
\]
TUNA specification

- **Xilinx Zynq SoC**
  - ARM Cortex-A9
  - Xilinx Kintex-7

- **Memory**
  - 1GB PS DDR3-SDRAM
  - For NVM: 8GB PL DDR3-SDRAM
Range: 1GB
Stride size: 4KB
\[ \text{Error} = 1.04\% \]
Latency Emulation: store

Range: 1GB
Stride size: 4KB
→ Error = 0.20%
HEAPO: Persistent Heap for Linux
Persistent Heap

Legacy heap

Persistent heap

Processes: fork()
Address Space: Persistent Area
Volatile

Processes: fork()
Address Space: Orthogonal Persistent!
Overall Structure of HEAPO

- Persistent heap layer
- Dynamic allocation/deallocation
- Filesystem-independent namespace
- Support pointer usage

HEAPO Library

- heapo_malloc("Object Storage A", 1836);
- heapo_free("Object Storage A", X);
- heapo_malloc("Object Storage A", 2824);
- heapo_free("Object Storage A", Y);

Kernel

- heapo_free()
- heapo_malloc()
- heapo_seg_free()
- heapo_seg_alloc()

Application

- heapo_malloc("Object Storage A", 1836);
- heapo_free("Object Storage A", X);
- heapo_malloc("Object Storage A", 2824);
- heapo_free("Object Storage A", Y);

HEAPO Library

Kernel

Application

- heapo_free()
- heapo_malloc()
- heapo_seg_free()
- heapo_seg_alloc()
Software Stack

Application

Library Interfaces
- heapo_create() / heapo_delete()
- heapo_map() / heapo_unmap()
- heapo_malloc() / heapo_free()
- heapo_set_prime_object() / heapo_get_prime_object()

HEAPO Library

Object Alloc/Dealloc Module
Object Storage Management Module
Name Table Cache

User

HEAPO Kernel Component

Kernel

System Calls
- sys_heapo_create() / sys_heapo_delete()
- sys_heapo_map() / sys_heapo_unmap()
- sys_heapo_seg_alloc() / sys_heapo_seg_free()

Name Space Management
Segment Allocation & Mapping
POS area Management
Object Storage & Object

- **Object storage**: Persistent memory region with unique name
- **Object**: Unit of memory allocation/deallocation
Usage Example

heapo_create("name")
heapo_malloc("name", size)
heapo_malloc("name", size)
heapo_malloc("name", size)
heapo_malloc("name", size)
heapo_free(address)
heapo_free(address)
exit()
fork()
heapo_map("name")
heapo_unmap("name")
H-KVLib: HEAPO-based Key-value Library

- Internal data structure: B-tree, list, hash
- Guarantee fail-safe atomicity (minimal logging)
- Support lock-based and STM-based concurrency control
Use case: Web Application

- **DRAM**
  - IO Buffer
  - Disk IO
  - Parsing (memory copy)
  - Rendering

- **DOM Tree**
  - Heap
  - Web Browser

- **Storage**
  - Web Browser
  - IO Buffer
  - Disk IO
  - Parsing (memory copy)
  - Rendering
Launching Web Applications with HEAPO

No Disk IO !
No Parsing !

Object Storage (NVRAM)

Web Browser

Rendering
HEAPo-based Key-value Store Performance

Insert (I), Lookup (L), Delete (D), Update (U)

BDB (Btree), BDB (Hash), HEAPo (Btree), HEAPo (Hash)

Operations per second:
- BDB (Btree)
- BDB (Hash)
- HEAPo (Btree)
- HEAPo (Hash)

Flash Memory Summit 2015
Santa Clara, CA
Fail Safe Atomicity Support in HEAPO

- Logging for key-value operations in H-KVLIB
- For persistence and ordering
  - `clflush` and `mfence` in x86
  - `dccmvac` in ARM
TUNA & HEAPO: HW/SW for NVRAM

- **TUNA platform** (http://www.opennvram.org)
  - OS support: linux & android
  - Applications and Library: HEAPO and SQLite for NVM

- **HEAPO** (https://github.com/ESOS-Lab/HEAPO)
  - HW support: x86 & ARM
  - Provide dedicated key-value library
Summary

- **HEAPo over TUNA**
  - Comprehensive Platform for developing software for NVM based system
  - All open sourced

- **TUNA platform** ([http://www.opennvram.org](http://www.opennvram.org))
  - OS support: linux & android

- **HEAPo** ([https://github.com/ESOS-Lab/HEAPo](https://github.com/ESOS-Lab/HEAPo)) for x86 and ARM
  - Provide dedicated key-value library
Collaborators

- Joint work with
  - CRZ Co.
  - Prof. Seungjoo Yoo at Seoul National University

- References
  - Lee et.al. “FPGA-based prototyping systems for emerging memory technologies,” IEEE International Symposium on Rapid System Prototyping (RSP), 2014